A New Class of Symbolic Abstract Neural Nets: Tissue P Systems

C. Martín-Vide[1], J. Pazos[2], G. Păun[1], and A. Rodríguez-Patón[2]

[1] Research Group on Mathematical Linguistics, Rovira i Virgili University
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain
{cmv,gp}@astor.urv.es
[2] Faculty of Computer Science, Polytechnical University of Madrid
Campus de Montegancedo, Boadilla del Monte 28660, Madrid, Spain
{jpazos,arpaton}@fi.upm.es

Abstract. Starting from the way the inter-cellular communication takes place by means of protein channels and also from the standard knowledge about neuron functioning, we propose a computing model called a *tissue P system*, which processes symbols in a multiset rewriting sense, in a net of cells similar to a neural net. Each cell has a finite state memory, processes multisets of symbol-impulses, and can send impulses ("excitations") to the neighboring cells. Such cell nets are shown to be rather powerful: they can simulate a Turing machine even when using a small number of cells, each of them having a small number of states. Moreover, in the case when each cell works in the maximal manner and it can excite all the cells to which it can send impulses, then one can easily solve the Hamiltonian Path Problem in linear time. A new characterization of the Parikh images of ET0L languages are also obtained in this framework.

1 Introduction

This paper can be seen at the same time as a contribution to neural networks (of a symbolic type), to membrane computing (with cells arranged in "tissues"), to finite automata networks (working not with strings, but with multisets of symbols), to multiset processing, to (distributed) automata and language theory. The motivation is two-fold: the inter-cellular communication (of chemicals, energy, information) by means of complex networks of protein channels (see, e.g., [1], [11]), and the way the neurons co-operate, processing impulses in the complex net established by synapses (see, e.g., [1], [2]).

The common mathematical model of these two kinds of symbol-processing mechanisms is the net of finite state devices, and this is the type of computing mechanisms we are going to consider: networks of finite-automata-like processors, dealing with symbols, according to local states (available in a finite number for each "cell"), communicating through these symbols, along channels ("axons") specified in advance. Note that the neuron modelling was the starting point of the theory of finite automata ([13], [10]), that symbol processing neural networks have a rich (and controversial) history (see [5] and its references), and that networks of string-processing finite automata have appeared in many contexts ([6], [9], [12], etc), but our models are different in many respects from all these previous models.

Having in mind the bio-chemical reality we refer to, a basic problem concerns the organization of the bunch of symbols available in each node, and the easiest and most natural answer is: no organization. Formally, this means that we have to consider *multisets* of symbols, sets with multiplicities associated with their elements. In this way, we need a kind of finite automata dealing with multisets of symbols, a topic which falls into an area of (theoretical) computer science not very much developed, although some recent (see, e.g., [7]), or not so recent (see, e.g., [4]) approaches can be found in the literature. Actually, most of the vivid area of membrane computing (P systems) [15] is devoted to multiset processing (details at `http://bioinformatics.bio.disco.unimib.it/psystems`).

The computing models we propose here, under the name of *tissue P systems*, in short, *tP systems*, consist of several *cells*, related by protein channels. In order to preserve also the neural intuition, we will use the suggestive name of *synapses* for these channels. Each cell has a state from a given finite set and can process multisets of *objects*, represented by symbols from a given alphabet. The standard rules are of the form $sM \rightarrow s'M'$, where s, s' are states and M, M' are multisets of symbols. Some of the elements of M' may be marked with the indication "go", and this means that they have to immediately leave the cell and pass to the cells to which we have direct links through synapses. This communication (transfer of symbol-objects) can be done in a replicative manner (the same symbol is sent to all adjacent cells), or in a non-replicative manner; in the second case we can send all the symbols to only one adjacent cell, or we can distribute them, non-deterministically. One more choice appears in using the rules $sM \rightarrow s'M'$: we can apply such a rule only to one occurrence of M (that is, in a sequential, *minimal* way), or to all possible occurrences of M (a *parallel* way), or, moreover, we can apply a maximal package of rules of the form $sM_i \rightarrow s'M_i', 1 \leq i \leq k$, that is, involving the same states s, s', which can be applied to the current multiset (the *maximal* mode). By the combination of the three modes of processing objects and the three modes of communication among cells, we get nine possible behaviors of our machinery.

A way to use such a computing device is to start from a given initial configuration (that is, initial states of cells and initial multisets of symbol-objects placed in them) and to let the system proceed until reaching a halting configuration, where no further rule can be applied, and to associate a result with this configuration. Because of the nondeterminism, starting from one given initial configuration we can reach arbitrarily many different halting configurations, hence we can get arbitrarily many outputs. Another possibility is to also provide *inputs*, at various times of a computation, and to look for the outputs related to them. Here we will consider only the first possibility, of *generative* tP systems, and the output will be defined by sending symbols out of the system. To this aim, one cell will be designated as the output one, and in its rules $sM \rightarrow s'M'$ we will also allow that symbols from M' are marked with the indication "out";

such a symbol will immediately leave the system, contributing to the result of the computation.

At the first sight, such a machinery (a finite net of finite state devices) seems not to be very powerful, e.g., as compared with Turing machines. Thus, it is rather surprising to find that tP systems with a small number of cells (two or four), each of them using a small number of states (resp., at most five or four) can simulate any Turing machine, even in the non-cooperative case, that is, only using rules of the form $sM \rightarrow s'M'$ with M being a singleton multiset; moreover, this is true for all modes of communication for the minimal mode of using the rules, and, in the cooperative case, also when using the parallel or the maximal mode of processing objects. When the rules are non-cooperative and we use them in the maximal mode, a characterization of Parikh images of ET0L languages is obtained, which completes the study of the computing power of our devices (showing that in the *parallel* and *maximal* cases we dot not get computational universality).

The above mentioned results indicate that our cells are "very powerful"; as their power lies in using states, hence in remembering their previous work, a natural idea is to consider tP systems with a low bound on the number of states in each cell. In view of the previously mentioned results, tP systems with at most 1, 2, 3, or 4 states per cell are of interest. We only briefly consider this question here, and we show that even reduced tP systems as those which use only one state in each cell can be useful: using such a net we can solve the Hamiltonian Path Problem in linear time (this is a direct consequence of the structure of a tP system, of the maximal mode of processing objects, and of the power of replicating the objects sent to all adjacent cells); remember that HPP is an NP-complete problem.

The power of tP systems with a reduced number of states per component remains to be further investigated. Actually, many other natural research topics can be considered, with motivations from automata and language theory (variants, power, normal forms), neural networks (learning, dynamic sets of neurons, dynamic synapses), computability (other NP-complete problems treated in this framework), dynamic systems (reachable configurations), etc.

2 Some Mathematical Prerequisites

The computability notions we use here are standard and can be found in many books, so we specify only some notations.

A *multiset* over a set X is a mapping $M : X \longrightarrow \mathbf{N}$; for $a \in X$, we say that $M(a)$ is the *multiplicity* of a in M. Here we work only with multisets over finite sets X. For two multisets M_1, M_2 over some set X we write $M_1 \subseteq M_2$ if and only if $M_1(a) \leq M_2(a)$ for all $a \in X$ (we say that M_1 is *included* in M_2). The *union* of M_1, M_2 is the multiset $M_1 \cup M_2 : X \longrightarrow \mathbf{N}$ defined by $(M_1 \cup M_2)(a) = M_1(a) + M_2(a)$, for all $a \in X$. If $M_1 \subseteq M_2$, then we also define the *difference* multiset $M_2 - M_1 : X \longrightarrow \mathbf{N}$ by $(M_2 - M_1)(a) = M_2(a) - M_1(a)$, for all $a \in X$. For $Y \subseteq X$ and M a multiset over X, we define the *projection* on Y by $pr_Y(M)(a) = \begin{cases} M(a), & \text{if } a \in Y, \\ 0, & \text{otherwise} \end{cases}$.

For a given alphabet V, V^* is the language of all strings over V, including the empty string, denoted by λ. The *Parikh mapping* associated with V is denoted by Ψ_V. A multiset M over an alphabet V can be represented by a string $w \in V^*$ such that $\Psi_V(w)$ gives the multiplicities in M of the symbols from V; obviously, all permutations of w are representations of the same multiset. For a family FA of languages, we denote by $PsFA$ the family of Parikh images of languages in FA. By CF, CS, RE we denote the families of context-free, context-sensitive, and recursively enumerable languages, respectively.

3 Tissue P Systems

We now pass to the definition of our variant of membrane (P) systems, which can also be considered as a model of a symbolic neural net. We introduce it in the general form, then we will consider variants of a restricted type.

A *tissue P system*, in short, a tP system, of *degree* $m \geq 1$, is a construct

$$\Pi = (E, \sigma_1, \ldots, \sigma_m, syn, i_{out}), \text{ where}$$

1. E is a finite non-empty alphabet (of *chemical objects*, but we also call them *excitations/impulses*);
2. $syn \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$ (*synapses* among cells);
3. $i_{out} \in \{1, 2, \ldots, m\}$ indicates the *output cell*;
4. $\sigma_1, \ldots, \sigma_m$ are *cells*, of the form $\sigma_i = (Q_i, s_{i,0}, w_{i,0}, P_i), 1 \leq i \leq m$, where:
   (a) Q_i is a finite set (of *states*);
   (b) $s_{i,0} \in Q_i$ is the *initial state*;
   (c) $w_{i,0} \in E^*$ is the *initial multiset* of impulses;
   (d) P_i is a finite set of *rules* of the form $sw \to s'xy_{go}z_{out}$, where $s, s' \in Q_i$, $w, x \in E^*$, $y_{go} \in (E \times \{go\})^*$ and $z_{out} \in (E \times \{out\})^*$, with the restriction that $z_{out} = \lambda$ for all $i \in \{1, 2, \ldots, m\}$ different from i_{out}.

A tP system as above is said to be *cooperative* if it contains at least a rule $sw \to s'w'$ such that $|w| > 1$, and *non-cooperative* in the opposite case.

Any m-tuple of the form $(s_1w_1, \ldots, s_mw_m)$, with $s_i \in Q_i$ and $w_i \in E^*$, for all $1 \leq i \leq m$, is called a *configuration* of Π; $(s_{1,0}w_{1,0}, \ldots, s_{m,0}w_{m,0})$ is the *initial configuration* of Π.

Using the rules from the sets P_i, $1 \leq i \leq m$, we can define *transitions* among configurations. To this aim, we first consider three *modes of processing the stimuli* and three *modes of transmitting excitations* from a cell to another one. Let us denote $E_{go} = \{(a, go) \mid a \in E\}$, $E_{out} = \{(a, out) \mid a \in E\}$, and $E_{tot} = E \cup E_{go} \cup E_{out}$. For $s, s' \in Q_i, x \in E^*, y \in E^*_{tot}$, we write

$$sx \Longrightarrow_{min} s'y \text{ iff } sw \to s'w' \in P_i, w \subseteq x, \text{ and } y = (x - w) \cup w',$$
$$sx \Longrightarrow_{par} s'y \text{ iff } sw \to s'w' \in P_i, w^k \subseteq x, w^{k+1} \not\subseteq x,$$

$$
\begin{array}{lll}
 & & \text{for some } k \geq 1, \text{ and } y = (x - w^k) \cup w'^k, \\
sx \Longrightarrow_{max} s'y & \text{iff} & sw_1 \rightarrow s'w'_1, \ldots, sw_k \rightarrow s'w'_k \in P_i, k \geq 1, \\
 & & \text{such that } w_1 \ldots w_k \subseteq x, y = (x - w_1 \ldots w_k) \cup w'_1 \ldots w'_k, \\
 & & \text{and there is no } sw \rightarrow s'w' \in P_i \text{ such that } w_1 \ldots w_k w \subseteq x.
\end{array}
$$

In the first case, only one occurrence of the multiset from the left hand side of a rule is processed (replaced by the multiset from the right hand of the rule, at the same time changing the state of the cell), in the second case a maximal change is performed with respect to a chosen rule, in the sense that as many as possible copies of the multiset from the left hand side of the rule are replaced by the corresponding number of copies of the multiset from the right hand side, while in the third case a maximal change is performed with respect to all rules which use the current state of the cell and introduce the same new state after processing the impulses.

We also write $sx \rightarrow_\alpha sx$, for $s \in Q_i, x \in E^*$, and $\alpha \in \{min, par, max\}$, if there is no rule $sw \rightarrow s'w'$ in P_i such that $w \subseteq x$. This encodes the case when a cell cannot process the current impulses in a given state (it can be "unblocked" after receiving new impulses from its ancestors).

The multiset w' from a rule $sw \rightarrow s'w'$ contains symbols from E, but also symbols of the form (a, go) (or, in the case of cell i_{out}, of the form (a, out)). Such symbols will be sent to the cells related by synapses to cell σ_i where the rule $sw \rightarrow s'w'$ is applied, according to the following modes:

- *repl*: each symbol a, for (a, go) appearing in w', is sent to each of the cells σ_j such that $(i, j) \in syn$;
- *one*: all symbols a appearing in w' in the form (a, go) are sent to one of the cells σ_j such that $(i, j) \in syn$, nondeterministically chosen; more exactly, in the case of modes *par* and *max* of using the rules, we first perform all applications of rules, and after that we send all obtained symbols to a unique descendant of the cell (that is, we do not treat separately the impulses introduced by each rule, but all of them in a package);
- *spread*: the symbols a appearing in w' in the form (a, go) are non-deterministically distributed among the cells σ_j such that $(i, j) \in syn$.

In order to formally define the transition among the configurations of Π we need some further notations. For a multiset w over E_{tot}, we denote by $go(w)$ the multiset of symbols $a \in E$ appearing in w in the form (a, go), and by $out(w)$ the multiset of symbols $a \in E$, appearing in w in the form (a, out). Clearly, $go(w)(a) = w((a, go))$ and $out(w)(a) = w((a, out)), a \in E$. Moreover, for a node i in the graph defined by syn we denote $ant(i) = \{j \mid (j, i) \in syn\}$ and $succ(i) = \{j \mid (i, j) \in syn\}$ (the ancestors and the successors of node i, respectively).

Now, for two configurations $C_1 = (s_1w_1, \ldots, s_mw_m), C_2 = (s'_1w''_1, \ldots, s'_mw''_m)$ we write $C_1 \Longrightarrow_{\alpha,\beta} C_2$, for $\alpha \in \{min, par, max\}, \beta \in \{repl, one, spread\}$, if there are $w'_1, \ldots, w'_m$ in E^*_{tot} such that $s_iw_i \Longrightarrow_\alpha s'_iw'_i$, $1 \leq i \leq m$, and

- for $\beta = repl$ we have $w''_i = pr_E(w'_i) \cup \bigcup_{j \in ant(i)} go(w'_j)$;

– for $\beta = one$ we have $w_i'' = pr_E(w_i') \cup \bigcup_{j \in I_i} go(w_j')$, where I_i is a subset of $ant(i)$ such that the set $ant(i)$ was partitioned into $I_1, \ldots, I_m$; at this transition, all non-empty sets of impulses of the form $\bigcup_{j \in I_k} go(w_j')$, $1 \leq k \leq m$, should be sent to receiving cells (added to multisets w_l'', $1 \leq l \leq m$);
– for $\beta = spread$ we have $w_i'' = pr_E(w_i') - go(w_i') \cup z_i$, where z_i is a submultiset of the multiset $\bigcup_{j \in ant(i)} go(w_j')$ such that $z_1, \ldots, z_m$ are multisets with the property $\bigcup_{j=1}^{m} z_j = \bigcup_{j \in ant(i)} go(w_j')$, and such that all $z_1, \ldots, z_m$ are sent to receiving cells (added to multisets w_l'', $1 \leq l \leq m$).

Note that in the case of the cell $\sigma_{i_{out}}$ we also remove all symbols $a \in E$ appearing in $w_{i_{out}}'$ in the form (a, out).

During any transition, some cells can do nothing: if no rule is applicable to the available multiset of impulses in the current state, then a cell waits until new impulses are sent to it from its ancestor cells.

A sequence of transitions among configurations of the tP system Π is called a *computation* of Π. A computation which ends in a configuration where no rule in no cell can be used, is called a *halting* computation. Assume that during a halting computation the tP system Π sends out, through the cell $\sigma_{i_{out}}$, the multiset z. We say that the vector $\Psi_E(z)$, representing the multiplicities of impulses from z, is *computed* (or *generated*) by Π. We denote by $N_{\alpha,\beta}(\Pi), \alpha \in \{min, par, max\}, \beta \in \{repl, one, spread\}$, the set of all vectors of natural numbers generated by a tP system Π, in the mode (α, β). The family of all sets $N_{\alpha,\beta}(\Pi)$, generated by all cooperative tP systems with at most $m \geq 1$ cells, each of them using at most $r \geq 1$ states, is denoted by $NtP_{m,r}(Coo, \alpha, \beta)$; when non-cooperative tP systems are used, we write $NtP_{m,r}(nCoo, \alpha, \beta)$ for the corresponding family of vector sets. When one (or both) of the parameters m, r are not bounded, then we replace it (them) with $*$, thus obtaining families of the form $NtP_{m,*}(\gamma, \alpha, \beta), NtP_{*,r}(\gamma, \alpha, \beta)$, etc.

We have 18 families of the form $NtP_{*,*}(\gamma, \alpha, \beta)$, but, as we will see below, not all of them are different.

4 An Example

Before investigating the power and the properties of tP systems, let us examine an example, in order to clarify and illustrate the previous definitions. Consider the rather simple tP system:

$$\begin{aligned}
\Pi_1 &= (\{a\}, \sigma_1, \sigma_2, \sigma_3, syn, 1), \\
\sigma_1 &= (\{s\}, s, a, \{sa \to s(a, go),\ sa \to s(a, out)\}), \\
\sigma_2 &= (\{s\}, s, \lambda, \{sa \to s(a, go)\}), \\
\sigma_3 &= (\{s\}, s, \lambda, \{sa \to s(a, go)\}), \\
syn &= \{(1,2), (1,3), (2,1), (3,1)\}.
\end{aligned}$$

The reader can easily check that we have:

$$N_{\alpha,repl}(\Pi_1) = \{(n) \mid n \geq 1\}, \text{ for } \alpha \in \{min, max\},$$

$$N_{par,repl}(\Pi_1) = \{(2^n) \mid n \geq 0\},$$
$$N_{\alpha,\beta}(\Pi_1) = \{(1)\}, \text{ for } \alpha \in \{min, par, max\}, \beta \in \{one, spread\}.$$

Indeed, in the non-replicative mode of communication, no further symbol is produced, hence we only generate the vector (1). In the replicative case, the symbols produced by the rule $sa \to s(a, go)$ from cell 1 are doubled by communication. When the rules are used in the parallel mode, then all symbols are processed at the same time by the same rule, which means that all symbols present in the system are doubled from a step to the next one, therefore, the powers of 2 are obtained. When the rules are used in the minimal mode, the symbols are processed or sent out one by one, hence all natural numbers can be obtained. In the maximal mode, we can send copies of a at the same time to cells 2 and 3, and outside the system, hence again any number of symbols can be sent out.

5 The Power of tP systems

The following relations are direct consequences of the definitions.

Lemma 1. (i) *For all* $1 \leq m \leq m', 1 \leq r \leq r', \gamma \in \{Coo, nCoo\}, \alpha \in \{min, par, max\}$, *and* $\beta \in \{repl, one, spread\}$, *we have:*

$$NtP_{m,r}(\gamma, \alpha, \beta) \subseteq NtP_{m',r'}(\gamma, \alpha, \beta) \subseteq NtP_{*,*}(\gamma, \alpha, \beta) \subseteq PsRE,$$
$$NtP_{m,r}(nCoo, \alpha, \beta) \subseteq NtP_{m,r}(Coo, \alpha, \beta).$$

(ii) *For all tP systems* Π, *cooperating or not, where each cell has at most one successor, and for all* $\alpha \in \{min, par, max\}$ *we have*

$$N_{\alpha,repl}(\Pi) = N_{\alpha,one}(\Pi) = N_{\alpha,spread}(\Pi).$$

As it is standard when considering a new computing device, we compare the power of tP systems with that of Turing machines and restricted variants of them. Refined classifications of the power of such machines are provided by the Chomsky and the Lindenmayer hierarchies. We start by considering the minimal mode of using the rules in a tP system, and this turns out to be computationally universal, a fact which makes natural the comparison with (Parikh images of) Chomsky families, in particular, $PsRE$. In a subsequent section we will consider the parallel and the maximal modes of using the rules, and this will make necessary the comparison with (Parikh images of) Lindenmayer families.

5.1 Comparison with Chomsky Families

Rather surprising, if we take into consideration the apparently weak ingredients of our models, when using the mode *min* of applying the rules, even the non-cooperative tP systems turn out to be computationally universal. (As expected, the same result holds true also when using cooperative rules, in all modes *min, par, max.*) In proving such results we try to keep as reduced as possible both the number of cells and the maximal number of states used by the cells.

Theorem 1. *$PsRE = NtP_{2,5}(\gamma, min, \beta)$ for all $\gamma \in \{Coo, nCoo\}, \beta \in \{repl, one, spread\}$.*

At the price of using two more cells, we can decrease the number of used states (the proof is omitted).

Theorem 2. *$PsRE = NtP_{4,4}(\gamma, min, \beta)$ for all $\gamma \in \{Coo, nCoo\}, \beta \in \{one, spread\}$.*

If we use cooperative rules, then we can further decrease both the number of cells and of states. Moreover, we can characterize $PsRE$ for all modes *min, par, max* of processing the impulses, and this completes the study of the cooperative case.

Theorem 3. *$PsRE = NtP_{2,2}(Coo, \alpha, \beta)$ for all $\alpha \in \{min, par\}, \beta \in \{repl, one, spread\}$.*

We do not know whether or not the results in Theorems 1, 2, and 3 are optimal in the number of cells and of states.

5.2 Comparison with Lindenmayer Families

The maximal mode of using the rules in a tP system resembles the parallel mode of rewriting the strings in an L system, and this makes the following results expected.

Theorem 4. (i) *$PsE0L \subseteq NtP_{1,2}(nCoo, max, \beta)$ for all $\beta \in \{repl, one, spread\}$.* (ii) *$PsET0L \subseteq NtP_{1,3}(nCoo, max, \beta)$ for all $\beta \in \{repl, one, spread\}$.*

For tP systems working in the *min* mode, we need further additional cells (and states) in order to simulate E0L and ET0L systems.

Theorem 5. *$PsE0L \subseteq NtP_{2,3}(nCoo, min, \beta)$ for all $\beta \in \{repl, one, spread\}$.*

In the case of ET0L systems we needed one more cell and one more state (but we do not know whether or not this result can be improved).

Theorem 6. *$PsET0L \subseteq NtP_{3,4}(nCoo, min, \beta)$ for all $\beta \in \{repl, one, spread\}$.*

Interestingly enough, the converse of assertion (ii) from Theorem 4 is also true, even in the following more general form (and this settles the study of modes *par* and *max*: they do not lead to computational universality).

Theorem 7. *$NtP_{*,*}(nCoo, \alpha, \beta) \subseteq PsET0L$, for all $\alpha \in \{par, max\}$ and $\beta \in \{repl, one, spread\}$.*

Together with assertion (ii) from Theorem 4 we get the following characterization of $PsET0L$, which precisely describes the power of the mode *max* in the non-cooperative case.

Theorem 8. *$NtP_{1,1}(nCoo, max, \beta) \subseteq NtP_{1,2}(nCoo, max, \beta) \subseteq NtP_{1,3}(nCoo, max, \beta) = NtP_{m,r}(nCoo, max, \beta) = NtP_{*,*}(nCoo, max, \beta) = PsET0L$, for all $m \geq 1, r \geq 3$.*

A more precise characterization of families $NtP_{m,r}(nCoo, par, \beta)$, $\beta \in \{repl, one, spread\}$, remains to be found (but we already know that such systems only generate Parikh images of ET0L languages).

6 Solving HPP in Linear Time

The architecture of tP systems and their way of working (especially the fact that in the maximal mode of using the rules we can process all impulses which may be processed in such a way that the same next state is obtained, irrespective which rules are used, and the fact that in the replicative mode one can send the same impulses to all successors of a cell) have an intrinsic computational power. More precisely, problems related to paths in a (directed) graph can be easily solved by a tP system, just by constructing a net with the synapses graph identical to the graph we deal with, constructing all paths in the graph with certain properties by making use of the maximal mode of applicating the rules and of the replicative communication, and checking the existence of a path with a desired property.

We illustrate this power of tP systems with the Hamiltonian Path Problem (HPP), which asks whether or not in a given directed graph $G = (V, U)$ (where $V = \{a_1, \ldots, a_m\}$ is the set of vertices, and $U \subseteq V \times V$ is the set of edges) there is a path starting in some vertex a_{in}, ending in some vertex a_{out}, and visiting all vertices exactly once. For simplicity, in what follows we assume that $a_{in} = a_1$ and $a_{out} = a_m$. It is know that the HPP is a NP-complete problem, hence it is one of the problems considered as intractable for the sequential computers (for the Turing machines).

Having a graph $G = (V, U)$ as above, we construct the tP system $\Pi = (E, \sigma_1, \ldots, \sigma_m, U, m)$, with

$$\begin{aligned}
E &= \{[z; k] \mid z \in V^*, 0 \leq |z| \leq m, 0 \leq k \leq m\},\\
\sigma_1 &= (\{s\}, s, [\lambda; 0], \{s[\lambda; 0] \to s([1; 1], go)\}),\\
\sigma_i &= (\{s\}, s, \lambda, \{s[z; k] \to s([zi; k+1], go) \mid z \in V^*, 1 \leq |z| \leq m-2,\\
&\qquad |z|_i = 0, 1 \leq k \leq m-2\}), \text{for each } i = 2, 3, \ldots, m-1, \text{ and}\\
\sigma_m &= (\{s\}, s, \lambda, \{s[z; m-1] \to s([zm; m], out) \mid z \in V^*, |z| = m-1\}).
\end{aligned}$$

It is easy to see that $N_{max,repl}(\Pi) \neq \emptyset$ if and only if HPP has a solution for the graph G: the paths in G grow simultaneously in all cells of Π, because of the *max* mode of using the rules (each cell has only one state, hence all rules can be used at the same time). Moreover, the cell σ_m can work only after $m-1$ steps and a symbol is sent out of the net at the step m. Thus, it is enough to watch the tP system at step m and if any symbol is sent out, then HPP has a solution, otherwise we know that such a solution does not exist. (Note that the symbol sent out describes a Hamiltonian path in G.)

References

1. B. Alberts et al., *Essential Cell Biology. An Introduction to the Molecular Biology of the Cell*, Garland Publ. Inc., New York, London, 1998.
2. M.A. Arbib, *Brains, Machines, and Mathematics*, second ed., Springer-Verlag, Berlin, 1987.
3. M.A. Arbib, *The Methaphorical Brain: An Introduction to Schemes and Brain Theory*, Wiley Interscience, 1988.
4. J.P. Banatre, D. LeMetayer, Gamma and chemical reaction model: ten years after, in vol. *Coordination Programming: Mechanisms, Models, and Semantics* (C. Hankin, ed.), Imperial College Press, 1996, 3–41.
5. D.S. Blank *et al* (24 co-authors), Connectionist symbol processing: Dead or alive?, *Neural Computing Surveys*, 2 (1999), 1–40.
6. C. Choffrut, ed., *Automata Networks*, *Lecture Notes in Computer Science*, 316, Springer-Verlag, Berlin, 1988.
7. E. Csuhaj-Varju, C. Martin-Vide, V. Mitrana, Multiset automata, *Multiset Processing* (C.S. Calude, Gh. Paun, G. Rozenberg, A. Salomaa, eds), *Lecture Notes in Computer Science*, 2235, Springer-Verlag, 2001.
8. A. Dovier, A. Policriti, G. Rossi, A uniform axiomatic view of lists, multisets, and sets, and the relevant unification algorithms, *Fundamenta Informaticae*, 36, 2-3 (1998), 201–234.
9. F. Gecseg, *Products of Automata*, Springer-Verlag, Berlin, 1986.
10. S.C. Kleene, Representation of events in nerve nets and finite automata, *Automata Studies*, Princeton Univ. Press, Princeton, N.J., 1956, 2–42.
11. W.R. Loewenstein, *The Touchstone of Life. Molecular Information, Cell Communication, and the Foundations of Life*, Oxford Univ. Press, 1999.
12. A. Mateescu, V. Mitrana, Parallel finite automata systems communicating by states, *Intern. J. Found. Computer Sci.*, to appear.
13. W.S. McCulloch, W.H. Pitts, A logical calculus of the ideas immanent in nervous activity, *Bull. Math. Biophys.*, 5 (1943), 115–133.
14. M. Minsky, *Computation: Finite and Infinite Machines*, Prentice-Hall, 1967.
15. Gh. Păun, Computing with membranes, *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143.